DOCUMENT RESUME

ED 274 315 IR 012 292

AUTHOR Yao, S. Bing; Hevner, Alan R.
TITLE A Guide to Performance Evaluation of Database Systems.
INSTITUTION Software Systems Technology, College Park, MD.
SPONS AGENCY National Bureau of Standards (DOC), Washington, D.C. Inst. for Computer Sciences and Technology.
REPORT NO NBS/SP-500-118
PUB DATE Dec 84
NOTE 58p.; Part of the Series, Reports on Computer Science and Technology.
AVAILABLE FROM Superintendent of Documents, U.S. Government Printing Office, Washington, DC 20402.
PUB TYPE Guides - Non-Classroom Use (055) -- Reports - Research/Technical (143)

EDRS PRICE MF01/PC03 Plus Postage.
DESCRIPTORS *Database Management Systems; *Databases; *Evaluation Criteria; *Evaluation Methods; Indexing; Performance; Reaction Time; Research Design
IDENTIFIERS *Benchmarking

ABSTRACT

Benchmarking is one of several alternate methods of performance evaluation, which is a key aspect in the selection of database systems. The purpose of this report is to provide a performance evaluation methodology, or benchmarking framework, to assist in the design and implementation of a wide variety of benchmark experiments. The methodology, which identifies criteria to be utilized in the design, execution, and analysis of a database system benchmark, has been applied to three different database systems representative of current minicomputer, microcomputer, and database machine architectures. This generalized methodology can apply to most database system designs. In addition to presenting a wide variety of possible considerations in the design and implementation of the benchmark, this methodology can be applied to the evaluation of either a single system with several configurations, or to the comparison of several systems. A summary of the report identifies the three principal phases of a database system benchmark--benchmark design, execution, and analysis--and notes that no generalized methodology can provide a complete list of considerations for the design of an actual experiment. Seventy references are listed. (DJR)

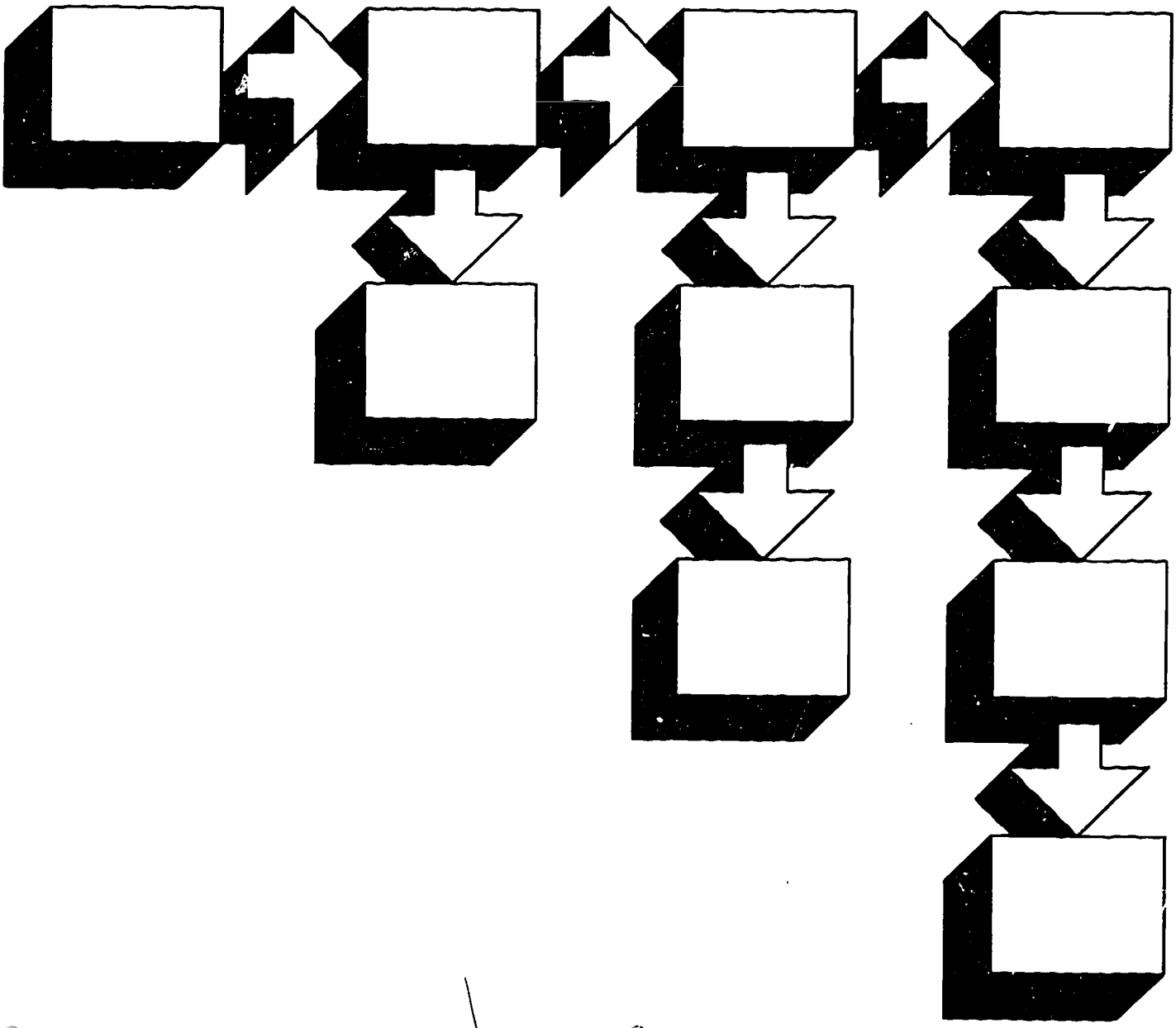
U.S. Department
of Commerce

National Bureau
of Standards

Computer Science and Technology

NBS Special Publication 500-118

A Guide to Performance Evaluation of Database Systems



The National Bureau of Standards[1] was established by an act of Congress on March 3, 1901. The Bureau's overall goal is to strengthen and advance the nation's science and technology and facilitate their effective application for public benefit. To this end, the Bureau conducts research and provides: (1) a basis for the nation's physical measurement system, (2) scientific and technological services for industry and government, (3) a technical basis for equity in trade, and (4) technical services to promote public safety. The Bureau's technical work is performed by the National Measurement Laboratory, the National Engineering Laboratory, the Institute for Computer Sciences and Technology, and the Center for Materials Science.

The National Measurement Laboratory

Provides the national system of physical and chemical measurement; coordinates the system with measurement systems of other nations and furnishes essential services leading to accurate and uniform physical and chemical measurement throughout the Nation's scientific community, industry, and commerce; provides advisory and research services to other Government agencies; conducts physical and chemical research; develops, produces, and distributes Standard Reference Materials; and provides calibration services. The Laboratory consists of the following centers:

- Basic Standards[2]
- Radiation Research
- Chemical Physics
- Analytical Chemistry

The National Engineering Laboratory

Provides technology and technical services to the public and private sectors to address national needs and to solve national problems; conducts research in engineering and applied science in support of these efforts; builds and maintains competence in the necessary disciplines required to carry out this research and technical service; develops engineering data and measurement capabilities; provides engineering measurement traceability services; develops test methods and proposes engineering standards and code changes; develops and proposes new engineering practices; and develops and improves mechanisms to transfer results of its research to the ultimate user. The Laboratory consists of the following centers:

- Applied Mathematics
- Electronics and Electrical Engineering[2]
- Manufacturing Engineering
- Building Technology
- Fire Research
- Chemical Engineering[2]

The Institute for Computer Sciences and Technology

Conducts research and provides scientific and technical services to aid Federal agencies in the selection, acquisition, application, and use of computer technology to improve effectiveness and economy in Government operations in accordance with Public Law 89-306 (40 U.S.C. 759), relevant Executive Orders, and other directives; carries out this mission by managing the Federal Information Processing Standards Program, developing Federal ADP standards guidelines, and managing Federal participation in ADP voluntary standardization activities; provides scientific and technological advisory services and assistance to Federal agencies; and provides the technical foundation for computer-related policies of the Federal Government. The Institute consists of the following centers:

- Programming Science and Technology
- Computer Systems Engineering

The Center for Materials Science

Conducts research and provides measurements, data, standards, reference materials, quantitative understanding and other technical information fundamental to the processing, structure, properties and performance of materials; addresses the scientific basis for new advanced materials technologies; plans research around cross-country scientific themes such as nondestructive evaluation and phase diagram development; oversees Bureau-wide technical programs in nuclear reactor radiation research and nondestructive evaluation; and broadly disseminates generic technical information resulting from its programs. The Center consists of the following Divisions:

- Inorganic Materials
- Fracture and Deformation[3]
- Polymers
- Metallurgy
- Reactor Radiation

[1]Headquarters and Laboratories at Gaithersburg, MD, unless otherwise noted; mailing address Gaithersburg, MD 20899.
[2]Some divisions within the center are located at Boulder, CO 80303.
[3]Located at Boulder, CO, with some elements at Gaithersburg, MD.

Computer Science and Technology

NBS Special Publication 500-118

A Guide to Performance Evaluation of Database Systems

Daniel R. Benigni, Editor
Center for Programming Science and Technology
Institute for Computer Sciences and Technology
National Bureau of Standards
Gaithersburg, MD 20899

Prepared by:
S. Bing Yao
Alan R. Hevner
Software Systems Technology, Inc.
7100 Baltimore Avenue, Suite 206
College Park, MD 20740

U.S. DEPARTMENT OF COMMERCE
Malcolm Baldrige, Secretary

National Bureau of Standards
Ernest Ambler, Director

Issued December 1984

Reports on Computer Science and Technology

The National Bureau of Standards has a special responsibility within the Federal Government for computer science and technology activities. The programs of the NBS Institute for Computer Sciences and Technology are designed to provide ADP standards, guidelines, and technical advisory services to improve the effectiveness of computer utilization in the Federal sector, and to perform appropriate research and development efforts as foundation for such activities and programs. This publication series will report these NBS efforts to the Federal computer community as well as to interested specialists in the academic and private sectors. Those wishing to receive notices of publications in this series should complete and return the form at the end of this publication.

Library of Congress Catalog Card Number: 84-601144

National Bureau of Standards Special Publication 500-118
Natl. Bur. Stand. (U.S.), Spec. Publ. 500-118, 54 pages (Dec. 1984)
CODEN: XNBSAV

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1984

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, DC 20402

TABLE OF CONTENTS

Page

FOREWORD 2

1. INTRODUCTION 3

2. PERFORMANCE EVALUATION TECHNIQUES 6

2.1 Analytic Modelling 6

2.1.1 Queueing Models 6
2.1.2 Cost Models 6

2.2 Simulation Modelling 7

2.3 Benchmarking 8

3. A BENCHMARK METHODOLOGY FOR DATABASE SYSTEMS 12

3.1 Benchmark Design 12

3.1.1 System Configuration 14
3.1.2 Test Data 14
3.1.3 Benchmark Workload 14
3.1.4 Experimental Design 15

3.2 Benchmark Execution 15

3.3 Benchmark Analysis 16

4. BENCHMARK DESIGN 17

4.1 System Configuration 17

4.1.1 Hardware Parameters 17
4.1.2 Software Parameters 17

4.2 Test Data 18

4.2.1 Constructing the Database 19
4.2.2 Database Size 21
4.2.3 Indexing 22

4.3 Benchmark Workload 23

4.3.1 Transactions 23
4.3.2 User-System Environment 24

4.3.3 Job-Scripts Model 25
4.3.4 Background Load 27

4.4 Experimental Design 28

4.4.1 Performance Measurement 28
4.4.2 Experimental Variables 29

5. BENCHMARK EXECUTION 33

5.1 Benchmark Initialization 33

5.1.1 Loading 33
5.1.2 Timing 34

5.2 Benchmark Verification 34

5.3 Benchmark Testing 36

6. BENCHMARK ANALYSIS 38

7. SUMMARY AND CONCLUSIONS 41

A GUIDE TO PERFORMANCE EVALUATION OF DATABASE SYSTEMS

Daniel R. Benigni, Editor

This guide presents a generalized performance analysis methodology for the benchmarking of database systems. The methodology identifies criteria to be utilized in the design, execution, and analysis of a database system benchmark. This generalized methodology can apply to most database system designs. In addition, presenting a wide variety of possible considerations in the design and implementation of the benchmark, this methodology can be applied to the evaluation of either a single system with several configurations, or to the comparison of several systems.

Key words: Benchmark execution; benchmark methodology; benchmark workload; database systems; DBMS; indexing; performance evaluation; query complexity; response time.

FOREWORD

This report is one of a continuing series of NBS publications in the area of data management technology. It concentrates on performance evaluation, which is a key aspect in the selection of database systems.

Benchmarking is one of several alternate methods of performance evaluation. It can be an expensive undertaking. However, this expense may be necessary for some applications, e.g., those involving large databases or where response time requirements are critical.

The purpose of this report is to provide a performance evaluation methodology, or benchmarking framework, to assist in the design and implementation of a wide variety of benchmark experiments. The methodology has been applied to three different database systems representative of current minicomputer, microcomputer, and database machine architectures. Detailed results can be found in [YAO 84].

Other NBS publications addressing various aspects of data management system selection include: FIPS PUB 77[NBS 80], NBS Special Publication 500-108 [GALL 84], and a forthcoming NBS publication on "Choosing a Data Management Approach." The advantages and disadvantages of benchmarking and other techniques for evaluating computer systems are discussed in NBS SP-500-113 [LETM 84].

References to commercial products as necessary to survey results of previous work on performance evaluation are contained in this guideline. In no case does this imply recommendation or endorsement by NBS.

1. INTRODUCTION

The rising popularity of database systems for the management of data has resulted in an increasing number of new systems entering the marketplace. As the number of available systems grows the difficulty in choosing the system which will best meet the requirements of a particular application environment also increases. Database systems have been implemented on many different computer architectures: mainframes, minicomputers, microcomputers, and as stand-alone database machines. The selection of a database system from among these varied alternatives requires a structured and comprehensive evaluation approach.

A complete evaluation methodology for database systems must integrate both feature analysis and performance analysis phases. The range of features and capabilities that a database system may support is very large. Feature lists for database systems have appeared in a number of articles [CODA 76, AUER 81, WEIS 81a, WEIS 81b, BARL 81, DATE 81, SU 81a, SU 81b, BROD 82].

A feature analysis performs two functions; it first serves as a winnowing process to eliminate those database systems which are completely unsuitable for answering the needs of a particular application; and second, it provides a ranking of the surviving candidate systems. Feature analysis is a widely used method of database system evaluation. It has a number of significant advantages over other methods of system evaluation.

1. A database system implementation is not required. Analysis is based upon documentation. Little or no system costs are involved in performing a feature analysis. This is critical for users with no system access.

2. Feature analysis provides a structured first cut for narrowing the range of potential database systems. A large number of systems can be evaluated effectively at one time. The result of a feature analysis should be a small number of candidate systems. Performance analysis, which is much more costly, need be performed on only this small number of systems.

3. The list of features evaluated can be customized to an organization's application environment and presented at the level of detail desired by the designer. Among these features are qualitative

aspects of a database system that cannot be quantified in terms of system performance. Examples include vendor support, documentation quality, security, "user friendliness", and reliability. Benchmark analysis cannot directly test the performance of these features. Thus, feature analysis remains the best method for their analysis.

In spite of these advantages, feature analysis should not be used in isolation to evaluate and select database systems. There are several reasons:

1. Feature analysis is a subjective exercise. Feature importance coefficients and the system support ratings needed in feature analysis are values which must be provided by a knowledgeable design expert. However, no two experts will come up with same values given the same application environment. At best, the feature analysis scores among different database systems should be viewed as rough indicators of the systems' applicability.

2. To obtain consistent scoring among different database systems the evaluator must be equally well acquainted with all systems. This places a great burden upon one person to acquire this knowledge. If instead, different persons grade different systems, then the scoring consistency problems increase because grading standards must be set and closely controlled.

3. The greatest disadvantage of feature analysis is that no true system performance is measured. Feature analysis is a paper exercise that cannot truly evaluate how a system will perform in an organization's application environment.

The limitations of feature analysis introduce the need for a more rigorous evaluation method that can provide objective, quantifiable differences among the candidate database systems. Performance analysis provides this type of evaluation.

The goals of performance analysis techniques are to model a database system's behavior and gather performance data. This is done to identify the system's strengths and weaknesses [LUCA 71, FERR 81]. Performance analysis has been utilized on database systems for two purposes. The first is to evaluate a single system to determine the best configuration, or running environment, for that system. For example, new system algorithms (e.g., file management [STON

83], query optimization [HEVN 79]) can be tested before actually implementing them in the system. In this way systems can be "tuned" for their most efficient operating condition. The other application of performance evaluation on database systems has been to study two or more database systems, thus providing a comparison of the systems' performance.

Section 2 presents an overview of past research on the performance evaluation of database systems. The purpose is to suggest that benchmark analysis is the most comprehensive technique for analyzing a single database system or comparing multiple database systems. An overview of the complete benchmark methodology is given in Section 3. The remainder of the report discusses in detail the design, execution, and analysis steps required in the benchmark methodology.

2. PERFORMANCE EVALUATION TECHNIQUES

The major methods of performance evaluation are Analytic Modelling, Simulation Modelling, and Benchmarking. A brief description of each method and a survey of previous work using the method for database system analysis is presented. The advantages and disadvantages of using each method are discussed.

2.1 Analytic Modelling

Analytic modelling represents a system by defining equations that relate performance quantities to known system parameters. The use of these equations allows a fast and accurate means to evaluate system performance. Two models that have been used predominantly to evaluate database system performance are queueing models and cost models.

2.1.1 Queueing Models.

The dynamic behavior of a database system can be viewed as a stochastic process and can be represented analytically as a queueing model [COMP 78]. A database system is modelled as a multiple resource system with jobs moving through the system demanding services from the resource stations. Queueing analysis provides the performance measures of system throughput, resource utilization, and job response time [MIYA 75]. The database system workload can be characterized by statistical parameters obtained from the datab requests [LEWI 76]. Because the database systems are usu ly quite complex, a queueing model normally can repres only a portion of its dynamic behavior. This is demonstrated clearly by the attempt to analytically model the scheduling of an IMS (IBM's Information Management System) database system in [GAVE 76, LAVE 76]. However, certain aspects of database system processing are conducive to queueing model analysis. For example, queueing models have been used to analyze concurrency control algorithms [SMIT 80, POTI 80] and data allocation in distributed database systems [COFF 81].

2.1.2 Cost Models.

Cost analysis has been an effective way of obtaining performance estimates for physical database structures. The performance measures most easily obtained by cost analysis are storage costs and average response time for queries.

Cost equations for inverted file systems have been developed in [CARD 73]. Generalized cost models have been pioneered in [YAO 74, 75, 77a, 77b] and further extended in [TEOR 76] and [BATO 82]. The cost model approach has been used to analyze the performance of query processing in various relational database systems [YAO 78, 79]. These models and cost functions have been useful for performing critical path analysis for database system applications. Hawthorne and Dewitt [HAWT 82] have developed cost models to evaluate query processing among different proposed database machine designs. A performance analysis of hierarchical processing in an IMS database system has been performed using cost models [BANE 80].

Analytic modelling has proven useful in many areas of database modelling. However, analytic models have some major disadvantages. Queueing models are inadequate to model the complete range of functionality found in a database system. Cost modelling fails to account for the dynamic behavior of the database system. For these reasons, analytic modelling has failed to receive wide acceptance as a tool for modelling database systems.

2.2 Simulation Modelling

Most real world systems are too complex to allow realistic models to be evaluated analytically. Simulation is the process of approximating the behavior of a system over a period of time. The simulation model is used to gather data as an estimate of the true performance characteristics of the system. Simulation modelling has been applied to database systems as illustrated in the following survey of representative work performed in this area.

A database system simulator, PHASE II, has been developed for the analysis of hierarchical data structures [OWEN 71]. PHASE II is an effective tool for evaluating hardware configurations, data structures, and search strategies. Another simulation model has been used to model the UNIVAC DMS-1100 database system [GRIF 75]. Hulten and Soderland designed the ART/DB simulation tool to investigate a multiprogrammed database system containing multiple CPUs [HULT 77]. This simulation tool is written in SIMULA and has an interactive interface.

A simulation model containing four modelling components (application program, database system, operating system, and hardware environment) was reported in [NAKA 75]. This simulation tool has two major sections: a definition section and

a procedure section. The definition section describes the system environment being simulated while the procedure section represents the software system using an instruction set prepared by the simulator. This type of programming interface allows a user to modify the system parameters while running a series of simulation programs. In another paper an IDS simulator is described [APPL 73]. A DBMS evaluation methodology through the integrated use of a limited prototype implementation for the DBMS design, a flexible measurement facility, and a predictive model based on the DBMS prototype was developed in [DEUT 79].

Similar to analytic models, simulation models are most often used to study specific types of database system processing. For example, recent simulation studies have analyzed optimal granule size for database locking [RIES 77, RIES 78], centralized versus decentralized concurrency control algorithms [GARC 78], and run-time schema interpretation on a network database system [BARO 82].

Although simulation modelling can be useful in systems which are too complex for analytic modelling methods, there are some disadvantages [LAW 82]. The major concern is the time and expense that are often necessary to develop a simulation model. Stochastic simulation models also produce only estimates of a model's "true" performance and the large volume of results returned by a simulation often creates a tendency to place more confidence in the results than may actually be warranted. As the simulation grows more complex, the difficulties in program verification increase correspondingly, making the validity of the results more difficult to determine.

2.3 Benchmarking

Benchmarking is used when a few database systems are to be evaluated and compared. Benchmarking requires that the systems be implemented so that experiments can be run under similar system environments. Benchmarks are costly and time consuming but provide the most valid performance results upon which database systems can be evaluated. In database benchmarking, a system configuration, a database, and a workload to be tested are identified and defined. Then tests are performed and results are measured and analyzed. The workload can be either representative of the planned application of the system (an application-specific benchmark) or designed to allow for an overall system evaluation (a general benchmark). Running the workload on several systems or several configurations of the same system will supply information which can be used to compare and evaluate the

separate systems or configurations.

Although simulation and analytic modelling have been useful in modelling aspects of database system behavior, benchmarking can apply to the complete database system functionality. While both simulation and analytic modelling are limited in the scope of their system testing, benchmarking offers the chance to evaluate the actual database system [GOFF 73]. Previous work on benchmarking database systems has been performed for two primary purposes. On a single database system, different implementation algorithms or different system configurations can be tested. For multiple database systems, the performance of the different systems on the same database and workload can be compared [DEAR 78].

Early database benchmark experiments concentrated on the comparison of candidate commercial systems for a particular application. For example, in [SPIT 77] an evaluation of three systems was described. The three systems tested were System 2000, GIM, and DMS-1100. All three systems ran on the UNIVAC CS-1100 computer system and were evaluated utilizing a specially designed monitoring system; the MITRE Performance Evaluation System, PES-1100. In [GLES 81], the benchmarking of several commercial systems to find the best system for the U.S. Public Health Service is described. One other early article [HILL 77] documented a performance analysis performed to select the best candidate system for NASA's Flight Planning System.

In the academic environment several studies have been performed on single database systems to evaluate performance and test enhancements. The System R access path optimizer is studied in [ASTR 80]. Benchmarks on the INGRES database system were reported in [YOUS 79, KEEN 81]. In [HAWT 79] detailed benchmarks were used to identify some possible enhancements to improve the performance of INGRES. The results of this study showed that dramatically increased performance on INGRES could be achieved by implementing a combination of extended memory, improved paging, and multiple processors. A later article [STON 83] described the effects of four enhancements to the INGRES system. The enhancements were dynamic compilation, microcoded routines, a special purpose file system, and a special purpose operating system. The results showed that while all four enhancements improved performance to some degree, the cost associated with the improvements were significant. While the compilation and file system tactics produced a high return and were relatively easy to accomplish, the microcode and special operating systems resulted in somewhat less of an improvement, and at a high cost. In [STON 82] and [EPST 80] the distributed INGRES database system was analyzed by

studying the performance results of queries.

More recently, comparisons of the performance of two or more database systems have been published. System Development Corporation has performed several comparison studies of benchmarks of database systems on a DEC VAX system [LUND 82, TEMP 82]. The first article compared ORACLE version 2.3.1 to the IDM-500 release 17. A report by Signal Technology, Inc. (STI), [SIGN 82] showed a comparison of STI's OMNIBASE to INGRES version 1.2/09. This article focused on specific test results and did not attempt to make an overall comparison of the two systems.

Bitton, DeWitt, and Turbyfill have described a customized database, a comprehensive set of queries and a methodology for systematically benchmarking relational databases [BITT 83]. In this study, testing was done on a synthetic database which was specifically designed for benchmarking. The benchmark included selections, projections, joins, aggregates, and updates on both the INGRES system, in two different configurations, and the IDM-500 database machine. The INGRES database systems, the 'university' and 'commercial' versions, were implemented on a VAX 11/750. The IDM-500 database machine was connected to a PDP 11/70 host and was studied both with and without a database accelerator. The purpose of the study was to test and compare the four configurations (two INGRES and two IDM). Although the original study was performed only in the single-user environment, a later paper [BORA 84] extended the project to the multiple user environment, by performing similar testing, with multiple users, on the IDM-500 database machine and ORACLE database system.

In [BOGD 83] an experiment in benchmarking a database machine was reported. The purpose of the paper was to present an approach to benchmarking database machines using a generated database. The paper describes a database generation tool which allows the user to build a synthetic (generated) database through an interactive interface. The description of the testing was quite general. The system configuration of the database machine was not fully described. The testing in this research was limited to the single-user case. The paper provided a summary of the testing results and numerous graphs plotting the results.

While benchmarking can be a useful and important technique for database system evaluation; designing, setting up, and running a benchmark is a difficult and time consuming task. In order to aid in the development and analysis of benchmarks it is essential that a generalized methodology be designed. While some work in this area has been done [TUEL

75, RODR 75, WALT 76, BITT 83, BOGD 83], no one methodology has provided the necessary robustness demanded of a generalized methodology. Most of the methods presented have been either tied to a limited number of systems, or have not rigorously addressed the possible testing variables and design characteristics necessary for a generalized methodology. In order to apply to many types of evaluation (e.g., general vs. specific, single system vs. many systems), a methodology must discuss many possible design and implementation features while providing guidance in the design of any benchmark experiment. In the next section an overview of the methodology is presented.

3. A BENCHMARK METHODOLOGY FOR DATABASE SYSTEMS

Managing a database requires a large, complex system made up of hardware, software, and data components. A benchmark methodology for database systems must consider a wide variety of system variables in order to fully evaluate performance. Each variable must be isolated as much as possible to allow the effects of that variable, and only that variable, to be evaluated. Because of the complex, interactive nature of database systems it is often very difficult, if not impossible, to do this. The benchmark methodology developed here enables a designer to identify the key variables of a database system to be evaluated. In this section a synopsis of the methodology is presented.

The benchmark methodology for database systems consists of three stages:

1. Benchmark Design - Establishing the system environment for the benchmark; involves designing the system configuration, test data, workload, and deciding on the fixed and free variables of the benchmark studies.

2. Benchmark Execution - Performing the benchmark and collecting the performance data.

3. Benchmark Analysis - Analyzing the performance results on individual database systems and, if more than one system is benchmarked, comparing performance across several systems.

Figure 3.1 illustrates the methodology as a flow chart. In this section, an overview of each phase is presented. The remainder of the report will discuss each phase in detail.

3.1 Benchmark Design

The design of a benchmark involves establishing the environment of the database system to be tested, and developing the actual tests to be performed. The four steps of the benchmark design phase are described below. For a comparative benchmark over several database systems, the benchmark design must be invariant over all systems.

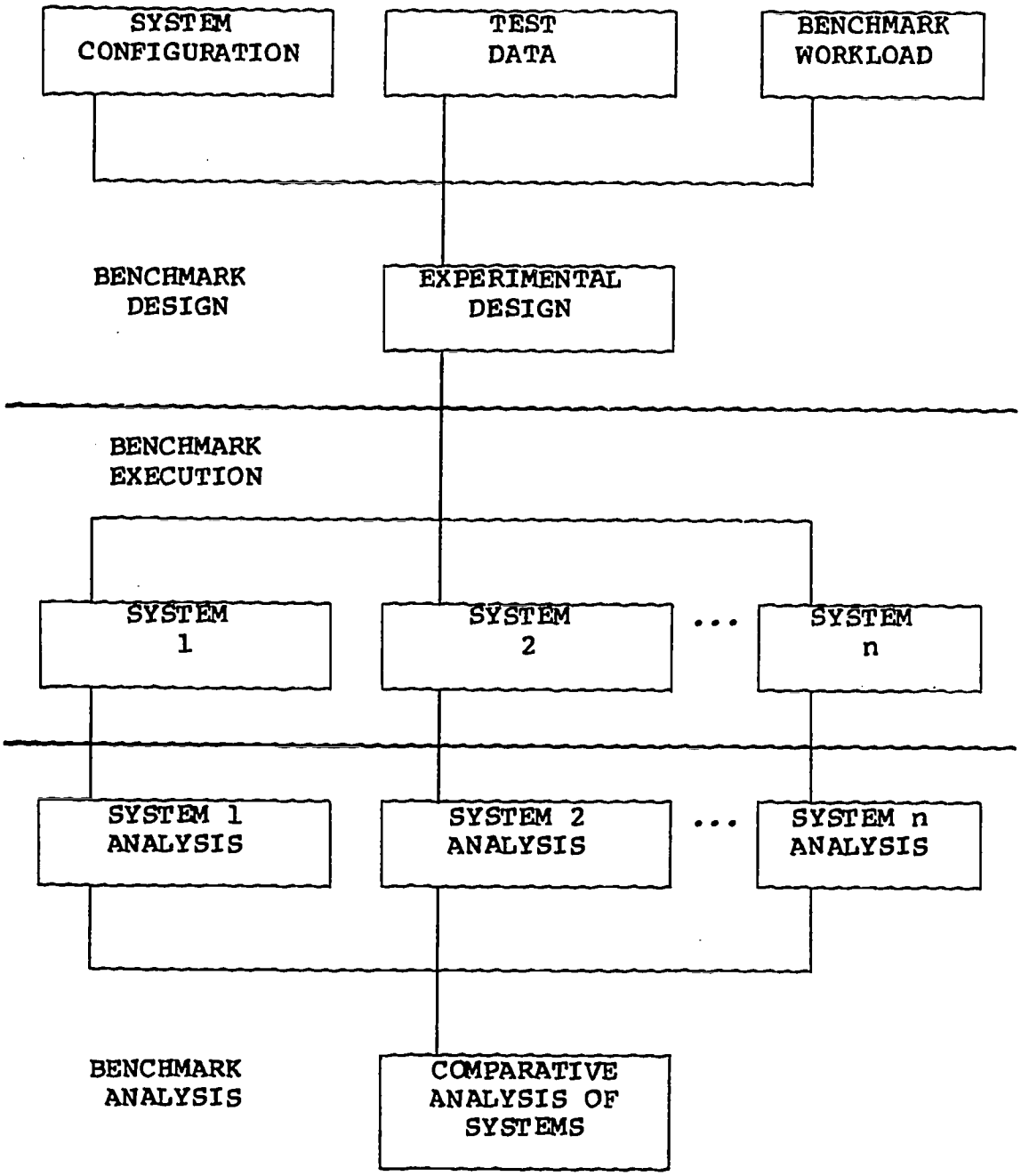


Figure 3.1: Database System Benchmark Methodology

3.1.1 System Configuration.

The hardware and software parameters, such as main memory size, the number and speed of the disk drives, operating system support for database system requirements, and load scheduling policies will be determined in this step. Often the hardware and software configuration is given. This is usually the case when the database system is to be added to an existing computing system. Also, many database systems can be installed on only one or very few types of operating systems. Cost is virtually always a factor and, for many applications, will be the primary determinant of which system configuration is actually chosen.

The parameters related to configuration that can be varied in the testing include maximum record length, the blocking factor of data in the storage system (e.g. the amount of data transferred by one disk access), the number of allowable indexes on relations, the maximum size and number of attribute values in an index, and the other types of direct access retrievals and their costs.

3.1.2 Test Data.

Among the parameters considered here are the test database, the background load, and the type and amount of indexing. The database on which the testing will be performed can be generated using one of two methods. The traditional method has been to use an already existing database, reformatting it for the benchmark needs. Recently, however, the approach of generating a synthetic database has been gaining popularity. Both techniques are discussed in Section 4.

3.1.3 Benchmark Workload.

A transaction load consists of several qualitative and quantitative aspects. Some of the qualitative aspects relating to transaction load are: the types of queries which occur (e.g., simple retrieval on a single relation or complex queries involving many files), the possible modes used for modification of the database (e.g., batch updates or interactive updates), the level of user-system interaction (e.g., on-line or batch access), and whether or not users commonly access the same data files. Some of the quantitative aspects of the transaction load include: the percentage of time that each type of database action is performed, the average amount of data returned to a user per transaction, the average number of users, and the number of users involved in each type of transaction. Therefore, the transaction load defines not only the number of users present in the system, but also the quality and degree of activity

introduced into the system by each user.

3.1.4 Experimental Design.

In this important phase of the benchmark design, parameters are selected to be varied in the benchmark testing. Values to be used for the parameters must also be defined. It is very important to choose values that, while within reason for the system being tested, push the system to its performance limitations. Among the parameters to be considered are database size, background load, number of indexes, query complexity, and number of simultaneous users.

It is also in this phase of the benchmark design that the criteria to be used for evaluation are considered. It is important to realize that the planned use of the system to be selected will have a definite relationship to the main measurement criteria on which the systems are evaluated. For example, if the system is expected to be used heavily and is likely to become CPU bound, system utilization or throughput would most likely be the main measurement criteria. On the other hand, if the system is more likely to be run under a light or moderate workload, response time would most likely be the most important criteria. The selection of measurement criteria for the experimental design is discussed in greater detail in Section 4.4.

3.2 Benchmark Execution

After the time-consuming and complex task of designing the benchmark is completed, the next step is to execute the experiments. It would make benchmarking a much less complicated task if the benchmark could be implemented exactly as designed on each individual system to be tested. In reality, this is seldom the case. Each system has its particular design and limitations to be considered. The benchmark has to be tailored to each specific system involved in the testing. Benchmark execution involves the steps of benchmark initialization, benchmark verification, and the actual benchmark tests. These steps are explained further in Section 5.

3.3 Benchmark Analysis

Benchmark experiments normally produce large amounts of output data that are too burdensome to evaluate. The final phase of a good benchmark experiment, therefore, must be a concise summary of the results obtained. This summary should point out the interesting results of the experiment and attempt to explain the reasons behind those results. A good summary will also present graphs relating testing parameters to response measures and matrices comparing results under varying variables. Benchmark analysis involves forming the raw performance data into graphs and tables that clearly illustrate observations and comparisons on the systems benchmarked. Benchmark analysis is fully described in Section 6.

Two types of benchmark analysis can be performed based upon the objectives of the benchmark testing.

1. Individual System Analysis - For each tested system, the data are analyzed to provide observations on the performance of the database system under varying system algorithms and conditions.

2. Comparative System Analysis - When multiple systems are being studied, performance data can be compared. This analysis should provide a basis to make statements as to critical comparison among several database systems.

4. BENCHMARK DESIGN

The benchmark design is made up of three areas which provide input to the final step of experimental design. These three areas; system configuration, test data, and the benchmark workload; as well as other factors involved in the experimental design, are discussed in this section.

4.1 System Configuration

System configuration consists of a wide variety of parameters which relate to both hardware and software. The hardware parameters include main memory size, the number and speed of disk drives, and data blocking. The software parameters include the operating system support, scheduling policies, and query optimization. Below is a list of some of the parameters considered in this phase and a brief discussion of each. A more detailed list of parameters can be found in [SU 81a, SU 81b].

4.1.1 Hardware Parameters.

1. Main memory size consists of the number of bytes of memory available for programs, user work areas, and input/output buffers.

2. Secondary storage consists of the number and type of disk drives. Parameters include disk capacity, seek time, and data transfer speed.

3. The configuration and speed of the communication lines in the system are important features that effect database system performance.

4. The speed of the CPU has an effect on the response time since most database systems experience CPU saturation conditions.

4.1.2 Software Parameters.

1. Memory page size has a direct effect on the locality of data allocation. Large page size enhances the clustering effect but requires larger buffer space. Unfortunately most systems do not permit the user to specify the page size as a parameter.

2. Indexing directly affects the data retrieval efficiency. Index parameters should include the type of index supported and any restrictions on the number of indexes permitted.

3. Operating system support and scheduling are often functions of the chosen operating system and are therefore difficult to test.

4. The query optimization scheme utilized by the software is not necessarily always the best method available. Comparison of an alternative method can often provide an interesting result. Although the query optimization algorithm is internal to a database system, some alternative algorithm's effect can be simulated through carefully arranged queries.

5. Database system control algorithms, such as concurrency control and recovery algorithms, may also be tested as to their effect on performance. Some database systems allow differing levels of control by setting system parameters. New control algorithms can be tested by adding the programs to the database system.

Many of the hardware and software parameters listed above are given, especially when the database system is to be added to an existing computer system. Often a database system can be installed on only one, or very few, types of operating systems. Therefore, testing is further constrained in regard to the selection of configuration parameters. It is usually difficult to vary database parameters such as buffer size and page size.

4.2 Test Data

A database is represented on a logical level by a conceptual schema that provides an overall view of the data and data relationships in the system. At this level the database is defined in terms of relations, records, attributes, and domains (using relational terminology). Hierarchical and network systems can be described in the appropriate data model terminology. At the physical level of representation the size and storage requirements of the database system

must be considered. In addition to the storage required to hold the data values in the database, access structures such as indexes must be included in the storage costs. Also, certain data may be duplicated for reliability or retrieval efficiency.

4.2.1 Constructing the Database.

One of the major considerations in any benchmark experiment is that of what test data will be used for the testing. The database used in the testing must be implemented on each of the candidate systems to be tested and after implementation must remain constant over all systems. There are basically two methods for obtaining a test database: using an already existing application database or developing a synthetic database.

Application Database

The traditional method has been the use of real data from an application database. By 'real data' is meant data that is being used, or has previously been used, for application purposes. If real data is to be used it must be formatted into the appropriate form for each system to be tested. If several systems are to be tested, the data must be formatted for each of the systems. If the database systems involved in the testing are not all of the same type (e.g. relational, hierarchical, or network), this formating can become a time consuming exercise in database design. Even when the systems involved are the same type, the loading and setting up of the database can produce unexpected problems. The use of real data, however, demonstrates database system performance on realistic application environments. This is clearly the best method when the evaluation is done to select a system for a known database environment.

Synthetic Database

The second method, the use of synthetic databases, has been gaining popularity in recent studies. When using this method, synthetic data is generated to make up a database which easily lends itself to benchmark testing. Attributes are allowed either integer or character values. Key attributes are assigned unique integer values. For example, for a relation with 10,000 tuples, the key attribute may take the values 0, 1, ... , 9999. The numbers can be scrambled using a random number generator. Other attributes are designed so that they contain non-unique values. The main purpose of these attributes is to provide a systematic way of modelling a wide range of selectivity factors. For example, a relation could be designed containing an attribute

with a uniform distribution of the values between 0 and 99. By utilizing the random number generator to create 5000 occurrences of this attribute into a relation, then queries can be easily designed with selectivities of 10%, 20%, ... , 90%, or any other percentage that is of interest in testing. Since the attribute has only 100 distinct values in the 5000 occurrences, 10% of the relation could be retrieved simply by running the following queries (using SQL):

```
SELECT <all>
FROM   <relation>
WHERE  <attribute>  < 10

       or

SELECT <all>
FROM   <relation>
WHERE  <attribute>  >  89
```

Such a design allows for much greater control over selectivity and can lead to a more precise result.

A major concern with the use of synthetic databases lies in the question of independence between attributes within a relation. In order to be certain that the attributes are truly independent each attribute within the relation must have an occurrence for each and every value of every other attribute in the relation. For example, a relation with two attributes (attr_1 and attr_2) containing values 0 through 1 and 0 through 2 respectively would have to contain the following records to demonstrate true attribute independence [ULLM 82].

Table 4.1: Independent Attributes

attr_1	attr_2
0	0
0	1
0	2
1	0
1	1
1	2

Obviously as the size of the relation grows, maintaining this independence leads to a very large relation.

While the use of a synthetic database does add a certain amount of control there are some drawbacks to using synthetic data. Synthetic relations fail to incorporate the complex data dependencies inherent to real data. For example, in a real database attributes within a relation, such as years of service and salary, have definite correlations which synthetic databases do not provide. Another factor to consider in choosing application vs. synthetic databases is the purpose of the testing. If the benchmark is being performed in order to select a system for a specific application it would obviously be preferable to test some data that would be used in the application. The use of a synthetic database is most suitable when designing a general benchmark over several database systems. However, the use of synthetic data takes away a certain measure of real world applicability present when using real data.

4.2.2 Database Size.

Database size is a key parameter and should be tested at various levels. Database sizes, 'small' to 'large', should be identified by studying the application system or the testing environment. In its use here 'large' means the largest database the application system is likely to use or the largest database available for testing. The term 'small' represents the point where the best performance is expected for the application. These points will often be identified during the benchmark and should be estimated initially.

Benchmark testing should begin on the smallest test database. By stepping to larger sizes, performance changes can be readily noted. When large performance gaps are noticed between database sizes additional database sizes may be tested in order to discover the point where the database size causes performance deterioration. Although availability of data may often dictate how large a database is used in

the benchmark, it is important to set the upper size at an adequate level to assure that possible applications will not normally exceed that level. By identifying a higher level than would normally be attained the effects of possible future growth can be evaluated.

The choice of the sizes of the databases to be tested will also be directly related to the system being tested and the resources available. If the testing is to be done on different configurations of one system, or comparing very similar systems, it is quite likely that each system could be tested on all of the sizes of the database to be tested. If, on the other hand, the testing is comparing aspects of systems of different sizes with differing capabilities, the database sizes tested on one system may be limited to a subset of the sizes tested on a larger system (e.g., micros and minis).

4.2.3 Indexing.

It is important to test the effects that indexing has on the performance of the systems being tested. Index levels should be set that will allow the systems to show differences in performance related to using the indexes. The transactions in the benchmark workload should be designed to highlight the potential performance benefits and costs of using indexes.

Some of the possible index levels that could be selected include:

0. No indexing. Studying results with no indexing provides a basis for comparison of the change in performance when utilizing indexes and is therefore essential.

1. Clustered indexes on key attributes. A clustered index is an index on an attribute whose data values are stored in sequential order. Clustered indexes on keys can be used effectively for retrieval and join operations.

2. Nonclustered indexes on secondary key attributes. Secondary key indexes, if used properly, will enhance the performance of queries that contain selection conditions using these keys.

3. Complete indexing. Indexes are placed upon all attributes in the database. The benefits of complete indexing must be weighted against its costs.

Database systems can also be tested for their ability to provide combined indexes. A combined index is one that ranges over two or more fields [STON 74]. Combined indexes can be defined on attribute groups that appear frequently together in queries.

4.3 Benchmark Workload

The benchmark workload is the heart of the benchmark experiment. While the system configuration and test data define the running environment, the benchmark workload defines the tests that will be run on this environment. By choosing a variety of transactions and then modelling user and system workloads by utilizing a job scripts model, a variety of benchmark parameters can be tested. The job scripts model is defined in Section 4.3.3. Transaction types that should be considered in the testing are discussed in the next section.

4.3.1 Transactions.

Each system, and each user on that system, is involved in a variety of transactions. A transaction is defined here as a well-defined user action on the database. In testing a database system, a variety of transaction types should be run. A benchmark should include the following types of transactions:

1. Single-Relation Queries - These queries involve only one relation. Testing on single relation queries should include queries on different sizes of relations, queries retrieving all or part of a relation, queries with and without indexes, and queries using aggregates and sorting functions such as "group-by" and "order-by".

2. Multi-Relation Queries - These queries involve more than one relation. Testing should include all of the variables discussed in the single-relation queries. A benchmark should also include testing on different join methods and varying the number of relations that are included in the query. Joins must be tested with and without indexes, and with different sequences of joining the relations.

3. Updates - Updates include functions such as modification, insertion, and deletion. Testing on updates must be performed carefully. These queries change the state of the database. If the effects of the updates are not removed, further testing on the database will not be performed on exactly the same data. Therefore the update effects must be removed from the database before further testing.

4.3.2 User-System Environment.

The next consideration in the design of the benchmark workload is the area of the user-system environment. This environment is a combination of factors including whether the system is on-line or batch, the frequency at which transactions enter the system, utilities offered by the system, and the programming interface.

There are basically two methods of executing transactions: batch and on-line.

1. Batch - A batch transaction is submitted and run with no interaction between a user and his job during processing.

2. On-line - An on-line transaction allows the user to interact with the program.

The frequency at which transactions enter the system is another factor that should be considered in the workload environment. Considerations regarding the amount of think time between transactions, the number and type of transactions, and the frequency of transactions on the system as the number of users grow, all must be taken into account. The job-scripts model defined in the next section is a convenient method of modelling these factors.

The utilities offered by the system are of prime concern to the database administrator and their functionality will be important to him. The utilities include creating, loading, and extending a table, creating an index, database dump, recovery, and checkpointing. User utilities may include sort packages, statistical routines, and graphics packages that can be interfaced with the database system.

4.3.3 Job-Scripts Model.

A job scripts model will be utilized to model the transaction load of the each user. By defining several distinct transaction loads and running them concurrently, a multi-user environment can be modelled.

A transaction load consists of several qualitative and quantitative aspects. Some of the qualitative aspects relating to transaction load are: the types of queries which occur (e.g., simple retrieval on a single relation or complex queries involving many relations), the possible modes used for modification of the database (e.g., update through menu modification, batch updates, updates in conjunction with queries), the level of user-system interaction (e.g., on-line vs. batch access), and whether or not users commonly access the same data files.

Some of the quantitative aspects of the transaction load include: the percentage of time that each type of database action is performed, the average amount of data returned to a user per transaction, the average number of users, and the number of users involved in each type of transaction. Therefore, the transaction load defines not only the number of users present in the system, but also the quality and degree of activity introduced into the system by each user.

Let a set of defined transactions be represented as T= { t(1), t(2), ... , t(n) } where t(i) represents the i-th transaction.

A job script represents a typical workload of an application user on the system. Let S be a set of pre-defined job scripts, s(i) , of the form:

s(i) = < t(j(1)), x(1), t(j(2)), x(2), ... , t(j(m)) >
where t(j(k)) is a transaction from the set T

and x(i) stands for the average think time found between successive user transactions. The x(i) parameters can also represent the interarrival times of transactions into the system. Job scripts can be designed to characterize a particular type of user in the system. For example, a database user may be described as retrieval intensive or update intensive. In either case a job script can be designed for that user by selecting transactions that best represent the user's typical processing.

The job script model consists of two defined sets: the job scripts, S = { s(1), s(2), ... , s(p) }, and the set of users, U = { u(1), u(2), ... , u(r) }. The workload for the system is defined by the number of users on the system and the assignment of users to job scripts. Each benchmark study is parameterized by the mix of job scripts that the systems execute. The assignment of users to one or more job scripts provides a very effective and clear way to characterize this job mix. An additional advantage of this job scripts model is that it can be easily implemented by a test generator program in the actual benchmark study.

A job script file for each user is read into a benchmark runner program that executes on the host computer system. The runner executes the transactions in job script order on the database system. This program also gathers performance data from the database system by recording statistics from hardware and software monitors. As an example, the following performance data can be collected on a database system for each transaction:

1. Parse time - The time required to parse the transaction and send it for execution.
2. Execution time - The time required to develop an access strategy for a transaction.
3. Time to first record - Time until the first result record is presented to the user.
4. Time to last record - Time until the last result record is presented to the user.
5. Number of records retrieved - The result size of the transaction; the size in bytes of the records should also be collected.

The benchmark runner algorithm can be outlined as follows:

```
Algorithm Runner:

Begin

  Read Job-script-file into trans-array until EOF;

  Open database system;

  While (trans-array not empty) do
```

```
    Read next transaction from trans-array;
    Parse transaction and send to database system;
    Execute transaction;
    Record time statistics on:
        time-to-parse
        time-to-execute
        time-to-first
        time-to-last;
    Record size statistics on:
        number of records in result
        size of result records;
    Print gathered statistics;

  End while;

  Close database system;

End of Algorithm.
```

For each benchmark test a job script is defined and executed on the different database systems to be tested. Statistics for each transaction in the job script are printed in a convenient format. For multi-user tests on the databases, multiple copies of the benchmark runner are run simultaneously on separate job scripts. Statistics are gathered for each job script.

4.3.4 Background Load.

The accurate evaluation of any system must take into account the type and amount of non-database work being performed by the host computer system. Based on typical system usage, as defined by the application, a number of non-database programs should be designed for background execution on the tested systems. These programs should be modelled using a job script in much the same manner as the user transaction loads. In this way measurements can be obtained for the effects of the background load on the database load, while the effects of the database load on the background load can also be measured. By enlisting the job script approach, parameters on the background load such as arrival rate of the programs, type of programs in the background load (e.g., CPU-bound vs, I/O bound) and priority given to programs in the background load, can be varied. Different background loads can be utilized in identifying system saturation points for the combined database and non-database system loads.

4.4 Experimental Design

Now that the running environment and possible alternatives for testing parameters have been defined, the parameters to be used in the testing should be selected. In order to properly evaluate the testing it will be necessary to set fixed values for most parameters, while testing others at various levels.

It is also at this stage of the benchmark design that the performance criteria to be used for evaluation are considered. The evaluation criteria selected are an essential key to understanding and correctly interpreting the benchmark results. In this section the selection of the measurement criteria will be discussed, as well as a review of the possible experimental parameters that can be varied in the testing.

4.4.1 Performance Measurement.

The relevant measures that may be considered for use in the performance evaluation include system throughput, utilization, and response time. Each of these will be discussed in the following paragraphs.

1. System throughput is defined as the average number of transactions (e.g., queries) processed per unit time. It stands to reason that as the number of queries on the system increases, approaching a saturation level, the system throughput will also increase. The throughput is a good indicator of the capacity and productivity of a system.

2. Utilization of a resource is the fraction of the time that the particular resource is busy. The main resources involved with processing database calls are the CPU, secondary storage, and the communication channels between them. Utilization, as with system throughput, is directly related to the number of transactions on the system at one time.

3. Response time can be taken to mean several different things. First, the response time could be considered as time-to-first-record. In other words, from the time the query enters the system until the time the first record in the response is returned. Another definition of response time could be time-to-last-record. This is from the time the query enters the system until the last record of the response is

available. Using this measurement would, of course, cause the size of the result to influence the performance measure. If the system is I/O bound the time to retrieve and deliver the entire response could eclipse the actual transaction processing time.

System throughput, utilization, and response time are all related in some sense. All three measurements tend to increase as the load on the system increases, but while a high system throughput, and a high utilization rate are perceived as desirable, a large response time carries a negative connotation.

Determining resource utilization and/or system throughput can be a very easy, or very difficult task depending upon the support offered by the system being tested. Some systems provide tools which offer easily accessible statistics while others require a great deal of user intervention using software tools to acquire the necessary information. Response time is usually the most readily available measure and is also the most user apparent. Because of these facts response time is the measurement utilized in most benchmarks.

The method used to perform the necessary calculations in determining response time is often a function of the support offered by the system being tested. For example, a database system running on an operating system which allows a flexible interface could support a very detailed timing algorithm, while a system with limited interfacing ability may require the use of a much more general timing method (e.g., setting a time at query input and again at query completion and recording the difference).

4.4.2 Experimental Variables.

A number of important tests can be performed in the benchmark by selecting and varying one or more dependent variables. The possible parameters that could be selected include:

1. Database Size - Several sizes relevant to the system being tested should be selected.

2. Query Complexity - Two factors are considered in determining query complexity. Greater complexity of the query predicate leads to increased parsing time and increases the potential for query optimization.

Within each query set, the predicate complexity is increased by adding additional conditions. A method of complexity classification has been developed by Cardenas [CARD 73]. The complexity of a query predicate increases in the following manner:

a. An atomic condition simply places a simple selection on a relation (e.g., Relation1.Att1 = '10').

b. An item condition is a disjunction (OR) of two atomic conditions on the same attribute (e.g., Relation1.Att1 = '10' OR Relation1.Att1 = '20').

c. A record condition is a conjunction (AND) of two item conditions (e.g., Relation1.Att1 = '10' AND Relation1.Att2 = 'ABC').

d. A query condition is a disjunction (OR) of record conditions (e.g., Relation1.Att1 = '10' OR Relation1.Att2 = 'ABC').

Second, the number of relations involved in a query indicates query complexity. More costly join operations are required when multiple relations are involved.

3. Records Retrieved - The response time of a transaction will depend greatly upon the number of records in the query result.

4. Order of Query Execution - The different database systems use internal memory as buffers for the storage of needed indexes and intermediate results during query execution. To test the effect of the buffer memory on the order of query execution, job scripts should be formed which consist of similar transaction loads executed sequentially. This will identify any efficiencies caused by buffering.

5. Indexing - Indexing should be tested at various levels. The use of at least three levels of indexes is recommended: no indexes, primary key indexes, and complete indexes.

6. Sorting - Sorting costs should be tested. One method of doing this is to add 'order by' clauses to the query sets. By comparing sorted and unsorted queries, the costs of sorting in the different database systems can be determined.

7. Aggregation Functions - Aggregation functions should be tested by adding 'count' or 'max' to the output list.

8. Number of Users - Multiple users contend for database system resources. This tends to increase the response time and increase the throughput. To study contention, tests should be run in which each user runs an identical job script. Other tests would include different combinations of job scripts. Multiple user tests will also test the database system's capabilities to control concurrency. Concurrent updates on the same data item will test the locking protocols of the systems.

9. Background Load - Tests should include runs varying the non-database jobs in the host computer system. The number and type of jobs in the background can be varied. Background jobs can be designed as CPU or I/O intensive jobs. Tests can determine the effect of these jobs on the performance of the database queries. By measuring the performance of the background jobs under different query loads, the effect of database jobs on the background jobs can also be studied. This is called a reverse benchmark.

10. Robustness - System performance should also be measured under controlled failure conditions in the system. This includes simulating the conditions of loss of power, disk failure, and software bugs. The capability to recover from these failures gracefully is an important feature of any database system. This includes the system's ability to recover on-going transactions, and to back out such transactions and request resubmission. Possible tests here include a deadlock test (it may or may not be easy to induce transaction deadlock) and disaster tests including a failed disk (demonstrate by powering down the disk), a failed system (power down the entire system), and an aborted transaction.

A benchmark may be either application- specific, meaning that it is intended primarily to evaluate the suitability of the candidate database systems for a particular application; or it can be more general, in which case the experiment is intended to perform an overall evaluation. When selecting parameters such as the types of transactions, number of users, and background load, the type of benchmark that is being performed will have a direct relation on the parameters selected. For example, when selecting

transactions to be run, if the benchmark is application-specific, the application environment would be studied and transactions modelled to duplicate this environment. In the more general case, a wide variety of transactions would be used with the intention of including the spectrum of actions typically performed in a user environment.

5. BENCHMARK EXECUTION

When the experiment has been formally defined the next step is to implement the design on each of the candidate systems. The actual execution of the benchmark can be broken into three phases: benchmark initialization, benchmark verification, and benchmark testing.

5.1 Benchmark Initialization

Before any testing can be performed the benchmark must first be initialized. During initialization the database is prepared for testing and the benchmark runner program is readied for use.

5.1.1 Loading.

The first step in preparing for the benchmark is to load the database into each of the test systems. While loading may seem like a simple step, this is often not the case. In most benchmark experiments the database is transferred from one system to each of the systems to be tested. When this is the case caution must be used to avoid inconsistencies in the databases. Any transfer of data between two systems requires the use of transfer protocols. When transferring a small amount of data the protocol between the systems may handle the details quite well. Test databases for benchmarking are usually quite large and transferring can be a very lengthy task. While not major factors, power failures, surges, and hardware malfunctions can create inconsistencies in the transfer and result in incorrect data.

If application data is used, the data must be reformatted into a usable form for the test system. The reformating of data can sometimes lead to unexpected problems which result in either the loading of unusable data, or not being able to load the data. If the data loaded into the test system turns out to be incorrect it may have to be dumped and the loading reinitialized with the necessary corrections. When loading large databases this can be a costly and time-consuming process.

Using a synthetic database does not eliminate the loading problems. If the database is generated in one system and subsequently transferred to the test systems, the same concerns as above apply. If, on the other hand, the database is generated on the test system, the consistency of the

test data must still be verified.

5.1.2 Timing.

It is also in this initialization phase that the timing algorithm in the benchmark runner is chosen, designed, and tested. In section 4.3 a detailed discussion of the possible measurement criteria to be returned by the timing algorithm is presented. In this phase the mechanism to be used is considered and tested. In the analysis of a single system, or compatible systems, the timing algorithm chosen should provide as detailed results as possible. Whether hardware monitors or software monitors are to be used, the initialization phase should allow for the tuning and testing of the tool.

When benchmarking several systems, the measurement criteria chosen must be implemented across all systems. The lack of ability to support a detailed timing algorithm by one system will sometimes limit the criteria that can be evaluated in comparisons. To insure that the timing method chosen is implementable across all test systems, the monitoring should be tested prior to any actual testing.

5.2 Benchmark Verification

Results obtained from benchmark experiments must be verified in order to be of any value. Three types of verification are discussed below.

1. Equivalence Verification. Each transaction is coded in the data manipulation language (e.g., SQL) of the different systems to be tested. It must be verified that the transactions are equivalent across all of the different systems to be tested. This equivalence can be tested by executing the transaction and checking that the results are correct and identical on all system. A more rigorous approach would be to prove, via semantic proving techniques, that the transactions are equivalent and will always produce identical results.

2. Optimum Performance Verification. In order to achieve fairness in the benchmarking, it should be verified that the transactions are coded so that the best performance can be realized in a typical configuration of each system. The existence and use of

access paths (e.g., index structures) should be as close to identical as possible in the different systems. The systems will be set up with normal amounts of processing power and storage as would be found in a typical configuration. In this way, each system will be shown in its 'best light'. A method of accomplishing this performance verification is to execute the transactions (after equivalence verification) on each system and collect performance results. Each vendor should be asked to evaluate the performance of his system and to suggest ways to tune the system for better performance.

3. Consistency Verification. During the execution of the benchmark experiments, a method for checking the consistency of performance results should be included in the experimental design. A consistency check consists of running a particular benchmark more than once and verifying that the performance results are consistent between the runs. While not all benchmark runs need to be duplicated, performing selected consistency checks will provide some assurance that the performance results are a function of the defined system environment.

The implementation of each job script must be verified both for correctness of its semantics and optimality of performance. When multiple database systems are being tested, the scope for verification measures is increased. In order to verify that the proper transactions are being used, the results of the queries (e.g., record counts, text of the returned fields) can be compared across database systems. Any discrepancies observed will trigger an inspection of the offending scripts. Common errors are: a missing qualification sub-clause, an improper sort order, or a syntactically correct but semantically incorrect operator in a selection clause.

Scripts may be changed during the initial check-out phase. A single benchmark experiment will run one or more scripts simultaneously on a target database system. Each script will be monitored so that the system response time which it experiences can be recorded. Simultaneously, statistics describing overall system throughput will be recorded. The choice of scripts to run together will be determined based upon the behavior in the database systems that are being tested. This is one area in which it is expected that preliminary evaluation of the benchmark results will feed back into the experiment, as new groups of scripts

will be suggested and tried out to probe specific performance features.

The details involving the transaction verification, as well as those involved with running the benchmark, will have a direct relation on how well the results from each system can be analyzed and compared to one another.

5.3 Benchmark Testing

Once the validity of the database has been assured, it is time to begin planning the benchmark runs on each of the systems to be tested. Differences between systems will necessitate differences in the approach to each implementation. For example, the limitations of a particular system's architecture may allow only a subset of the experiment to be applied. So, while the methodology flowchart presented earlier shows a clean interface between the design and execution phases, in reality this is rarely, if ever, the case when implementing the benchmark on more than one system. It will often be necessary to return to the design phase in order to restrict or revise the planned testing on each particular system to be tested.

Any general design of a benchmark will encounter problems specific to a given environment. The variations of the hardware and operating system environments, as well as the particular database system, will cause the experiment to vary from its original design. In an application-oriented benchmark any system which lacks the functionality to perform all of the desired operations presumably has already been eliminated during the features analysis phase. However, a general benchmark may include a diverse set of systems, some of which cannot perform all of the tests, or may perform limited versions of them.

The hardware involved can cause departures from what is desired. If, for instance, there is a limited amount of main storage available, the operating system and the database system taken together may preclude the testing of any type of background load altogether. Also, limited amounts of main storage reduce the possible complexity of the database system. Interaction between the various hardware/ operating system/ database system components can have deleterious effects. For example, the conflicting seeks used by an operating system and the DBMS system software are shown to have degraded performance of benchmark tests run by System Development Corporation [LUND 82].

In the case of a small computer database system, the limited amount of main storage may not allow the database system enough space to have the code to implement all of the functions specified in the scripts. Aggregate functions (e.g., MAX, MIN, AVERAGE) are not present in the query languages of many database systems. More exotic (but useful) operations such as an 'outer join' are present in very few of today's systems.

The benchmark tests must be carefully monitored during their execution, and as knowledge is gained from the experiments, it is expected that the original experiment will be redesigned to take advantage of that knowledge. Hence, running the benchmarks is not a completely 'cut-and-dried' task. Most of the benchmark involves running scripts against each of the target systems, while varying individual parameters of the systems. After each run, the results will be scanned to verify that embedded consistency checks have not uncovered any anomalies.

Once the experiments are running smoothly, the effort of interpreting the data will begin, and henceforth, the processes of gathering data and interpreting it can proceed in parallel. The results obtained should suggest new combinations of parameters and scripts, or variations on old ones, which will be run to probe specific aspects of a system's performance.

6. BENCHMARK ANALYSIS

The final phase of benchmarking is the analysis of results. Benchmark testing often produces large amounts of raw data which must be condensed and analyzed. Evaluation of the data generated during benchmarking must begin before the tests have been completed. This provides feedback during the testing by suggesting which types of experiments need to be repeated in more detail, or should be extended in some way. Summarizing the meaningful information from these results and discussing them in a report form is a key step in the benchmark testing. Explanations are provided for any significant findings and graphs and tables showing the comparison of results are included. Analyses are made on both individual systems, comparing results by varying parameters, and between systems, comparing one's results to the other's results. The following section discusses some of the possible comparisons to be drawn when analyzing a single system or when comparing several systems.

Each test parameter should be evaluated in as isolated an environment as possible so that the results can be directly attributable to the current configuration of parameters. A matrix of parameters to be evaluated (e.g., database size, background load, etc.) should be designed and performance benchmarks run for each combination.

When the testing is complete, results of each system are thoroughly analyzed. The parameters defined as those to be varied in the testing are to be monitored, and their behavior summarized. Graphs are utilized to demonstrate these behaviors. Graphs provide a clear, concise synopsis of the relationship between parameters and should be utilized frequently in the analysis phase.

When evaluating a single system, a variety of comparisons should be studied in order to identify interesting results. Below are several possible comparisons and effects that should be included in the analysis.

1. Response Time vs. Query Complexity. Under normal conditions, the relationship between these two parameters should be an increase in response time as the query becomes more complex due to an increase in parsing and execution. This is shown when considering the time-to-first-record response time statistic.

2. Response Time vs. Records Retrieved. This relationship should provide an interesting result regarding the time relationship when retrieving increasing numbers of records.

3. Response Time vs. Indexing. Indexing should have a positive effect (decreasing response time) for most queries but this is not always the case. Some indexing may actually cause the response time to increase if the query accesses a large percentage of the relation (high hit ratio).

4. Buffering. Efficient use of buffer space can sometimes lead to improved performance. However, this requires the use of special buffer management algorithms which are not implemented in most database systems.

5. Sorting. Sorting can be an expensive process. A test showing the difference in a query when it is sorted vs. when no sorting is required can identify the system's strength in this area.

6. Aggregation. The resulting increase in cost as the number of records retrieved increases using an aggregate function should be documented.

7. Multi-user Results. The effects of multi-users on the database are an important and realistic testing parameter. The amount of increase in response time as the number of users on the system increases should be calculated and graphed.

8. Background Load Results. In much the same manner as the multi-user environment, the background load can have a dramatic effect on the response time. As the background load increases the resulting increase in response time on the database system workload should be monitored.

9. Reverse Benchmark. The database workload will have an effect on the performance of applications on host computer systems. An analysis of this effect should be performed.

Performance saturation points will occur when a system shows a marked decrease in performance based upon a resource becoming overloaded. System saturation points should be identified and plotted for each of the systems tested. The saturation level will be a function of the number of users,

types of workload, background load, and other tested parameters. Therefore, an explanation of the saturation points on each configuration and some comments regarding the level are necessary to backup the graphs.

Finally, the end product of any benchmark experiment should be a report which summarizes the interesting findings of the testing, discusses the reasons behind the findings, and draws comparisons between the systems tested. If any possible solutions exist to problems identified in the benchmark they should be recommended in writing. This report should stress general, rather than specific, results and therefore provide an overall evaluation of the systems.

7. SUMMARY AND CONCLUSIONS

This report has presented a general methodology for the benchmarking of database systems. Previous projects on database system benchmarks, surveyed in Section 2, have identified many different factors that influence database performance. The objective in this report has been to describe a framework into which these many database system parameters can be fitted. Three principal phases of a database system benchmark have been identified.

1. Benchmark design includes the design of the system configuration, the test data, and the benchmark workload. These parameters are controlled in the experimental design of the benchmark. Performance measures and the means to gather the performance statistics are selected.

2. Benchmark execution implements the design on one or more database systems. This phase requires strict verification procedures and may involve feedback for improving the benchmark design.

3. Benchmark analysis is the phase in which the raw performance data is studied and observations and conclusions are made. Single system analysis and multiple system comparisons form the result of the benchmark.

The design of a benchmark methodology is a complex task. Previous work on database system benchmarks has been applied to selected cases of interest. In this methodology an attempt has been made to present a framework in an orderly, top-down fashion to assist the designer of a benchmark experiment in the design and implementation of a benchmark.

In attempting to design a generalized, standard approach to benchmarking, the complexity of the actual task of designing a specific benchmark must be taken into account. No generalized methodology can provide a complete list of considerations for the design of an actual experiment. Instead, the methodology can only provide the user with as comprehensive a list of system parameters as possible. Each experiment and each system has its own characteristics and constraints. While the methodology will help the designer by providing a comprehensive framework for the benchmark, it is the designer's task to fit the particular aspects of each database system, application environment, and operating

constraints into a viable benchmark study.

REFERENCES

[APPL 73] Applied Computer Research, "IDS Simulator - Functional Description," Internal Report, Jan. 1973.

[ASTR 80] Astrahan, M., Schkolnick, M. and Kim W. "Performance of the System R Access Path Selection Mechanism," Proceedings IFIP Conference, 1980.

[AUER 81] Auerbach Publishers Inc. Practical Data Base Management, Reston Publishing Company, 1981.

[BANE 80] Banerjee, J., Hsiao, D. and Ng, F. "Database Transformation, Query Translation, and Performance Analysis of a New Database Computer in Supporting Hierarchical Database Management," IEEE Transactions on Software Engineering, Vol. SE-6, No. 1, January 1980.

[BARL 81] Barley, K. and Driscoll, J. "A Survey of Data-Base Management Systems for Microcomputers," BYTE, November 1981.

[BARO 82] Baroody, A. and DeWitt, D. "The Impact of Run-Time Schema Interpretation in a Network Data Model DBMS," IEEE Transactions on Software Engineering, Vol. SE-8, No. 2, March 1982.

[BATO 82] Batory, D. "Optimal File Designs and Reorganization Points," ACM Transactions on Database Systems, Vol. 7, No. 1, March 1982.

[BITT 83] Bitton, H. Dewitt, D., and Turbyfill, C. "Benchmarking Database Systems: A Systematic Approach," Computer Sciences Department Technical Report #526, Computer Sciences Department, University of Wisconsin, January 1983.

[BOGD 83] Bogdanowicz, R., Crocker, M., Hsaio, D., Ryder, C., Stone, V., and Strawser, P. "Experiments in Benchmarking Relational Database Machines," Proceedings of the Third International Workshop on Database Machines, Munich, West Germany, Sept. 1983.

[BORA 84] Boral, H., and Dewitt, D. "A Methodology for Database System Performance Evaluation," Computer Sciences Technical Report #532, Computer Sciences Department, University of Wisconsin, January 1984.

[BROD 82] Brodie, M. and Schmidt, J. "Final Report of the

ANSI/X3/SPARC DBS-SG Relational Database Task Group," Document SPARC-81-690, ACM SIGMOD Record, July 1982.

[CARD 73] Cardenas, A. "Evaluation and Selection of File Organization - A Model and System," Communications of the ACM, Vol. 16, No. 9, 1973.

[CODA 76] CODASYL Systems Committee, Selection and Acquisition of Data Base Management Systems, ACM, New York, 1976.

[COFF 81] Coffman, E., Gelenbe, E. and Plateau, B. "Optimization of the Number of Copies uin a Distributed Data Base," IEEE Transactions on Software Engineering, Vol. SE-7, No. 1, January 1981.

[COMP 78] Computer Surveys, Special Issue: Queueing Network Models of Computer System Performance, Vol. 10, No. 3, September 1978.

[DATE 81] Date, C. An Introduction to Database Systems, Third Edition, Addison-Wesley Inc., 1981.

[DEAR 78] Dearnley, P. "Monitoring Database System Performance," The Computer Journal, Vol. 21, No. 1, 1978.

[DEUT 79] Deutsch, Don "Modeling and Measurement Techniques for the Evaluation of Design Alternatives in the Implementation of Database Management Software," NBS Special Publication 500-49, U.S. Dept. of Commerce, National Bureau of Standards, July 1979.

[EPST 80] Epstein, R. and Stonebraker, M. "Analysis of Distributed Data Base Processing Strategies," Proceedings of the 6th VLDB, Montreal, Canada, 1980.

[FERR 78] Ferreri, D. Computer Systems Performance Evaluation, Prentice-Hall Inc., 1978.

[GALL 84] Gallagher, L.J., and Draper, J.M. Guide on Data Models in the Selection and Use of Database Management Systems, NBS Special Publication 500-108, January 1984.

[GARC 79] Garcia-Molina, H. "Performance of Update Algorithms for Replicated Data in a Distributed Database," Report STAN-CS-79-744, Stanford University, Dept. of Computer Science, 1979.

[GAVE 76] Gaver, D., Lavenberg, S. and Price, T. "Exploratory Analysis of Access Path Length Data for a Data Base Management System," IBM Journal of Research and Development, Vol. 20, No. 5, Sept. 1976.

[GLES 81] Gleser, M., Bayard, J., and Lang, D. "Benchmarking for the Best," Datamation, May 1981.

[GOFF 73] Goff, N. "The Case for Benchmarking," Computers and Automation. May 1973.

[GRIF 75] Griffith, W. "A Simulation Model for UNIVAC DMS-1100 - More Than Just a Performance Evaluation Tool," Proceedings of the Symposium on the Simulation of Computer Systems, Boulder, Colorado, 1975.

[HAWT 79] Hawthorn, P. and Stonebraker, M. "Performance Analysis of a Relational Data Base Management System," Proceedings of the ACM SIGMOD Conference, Boston, 1979.

[HAWT 82] Hawthorn, P. and DeWitt, D. "Performing Analysis of Alternative Database Machine Architectures," IEEE Transactions on Software Engineering, Vol. SE-8, No. 1, January 1982.

[HEVN 79] Hevner, A. R. "The Optimization of Query Processing on Distributed Database Systems," Ph.D. Thesis, Database Systems Research Center Report DB-80-02, Department of Computer Science, Purdue University, December 1979.

[HILL 77] Hillman, H. "A Performance Analysis of Several Candidate Computers for NASA's Flight Planning System," MTR-4599, The MITRE Corporation, March 1977.

[HULT 77] Hulten, C. and Soderlund, L. "A Simulation Model for Performance Analysis of Large Shared Data Bases," Proceedings Third VLDB Conference, Tokyo, 1977.

[KEEN 81] Keenan, M., "A Comparative Performance Evaluation of Database Management Systems", EECS Dept., University of California, Berkeley, CA, 1981.

[LAVE 76] Lavenberg, S. and Shedler, G. "Stochastic Modelling of Processor Scheduling with Application to Data Base Management Systems," IBM Journal of Research and Development, Vol. 20, No. 5, Sept. 1976.

[LAW 82] Law, A. and Kelton, W. Simulation Modelling and Analysis, McGraw-Hill Book Company, 1982.

[LETM 84] Letmanyi, Helen "Assessment of Techniques for Evaluating Computer Systems for Federal Agency Procurement," NBS Special Publication 500-113, U.S. Dept. of Commerce, National Bureau of Standards, March 1984.

[LEWI 76] Lewis, P. and Shedler, G. "Statistical Analysis of Nonstationary Series of Events in a Data Base System," IBM Journal of Research and Development, Vol. 20, No. 5, Sept. 1976.

[LUCA 71] Lucas, H. "Performance Evaluation and Monitoring," Computer Surveys, Vol. 3, No. 3, September 1971.

[LUND 82] Lund, E. and Kameny, I., "Preliminary Comparison of Performance Results of ORACLE Release 2.3.1 on VAX/VMS with IDM-500 Release 17 on VAX/UNIX", SDC document SP-4158/000/00.

[MIYA 75] Miyamoto, I. "Hierarchical Performance Analysis Models for Database Systems," Proceedings First VLDB Conference, Farmingham, 1975.

[NAKA 75] Nakamura, F., Yoshida, I. and Kondo, H. "A Simulation Model for Data Base System Performance Evaluation," Proceedings NCC, 1975.

[NBS 80] NBS Guideline for Planning and Management of Database Applications, FIPS PUB 77, September 1980.

[OWEN 71] Owen, P. "PHASE II: A Database Management Modeling System," Proceedings of the IFIP Conference, 1971.

[POTI 80] Potier, D. and Leblanc, P. "Analysis of Locking Policies in] Database Management Systems, Communications of the ACM, Vol. 23, No. 10, October 1980.

[RIES 77] Ries, D. and Stonebraker, M. "Effects of Locking Granularity in a Database Management System," ACM Transactions on Database Systems, Vol. 2, No. 3, September 1977.

[RIES 79] Ries, D. and Stonebraker, M. "Locking Granularity Revisited," ACM Transactions on Database Systems, Vol. 4, No. 2, June 1979.

[RODR 75] Rodriguez-Rosell, J. and Hilderbrand, D. "A Framwork for Evaluation of Data Base Systems,"

Research Report RJ 1587, IBM San Jose, 1975.

[SIGN 82] Signal Technology, Inc., "OMNIBASE Test Results", Internal Report, 1982.

[SMIT 80] Smith, C. and Browne, J. "Aspects of Software Design Analysis: Concurrency and Blocking," Proceedings of the Performance 80 Symposium, Toronto, 1980.

[SPIT 77] Spitzer, J. and Patton, J. "Benchmark Analysis of JSC's Database Management Systems," Proceedings Spring 1977 ASTUTE Conference.

[STON 74] Stonebraker, M. "The Choice of Partial Inversions and Combined Indices," Jounal of Computer Information Sciences, Vol. 3, No. 2, 1974.

[STON 80] Stonebraker, M. "Retrospective on a Database System," ACM Transaction on Database Systems, Vol. 5, No. 2, 1980.

[STON 82] Stonebraker, M. et al. "Performance Analysis of Distributed Data Base Systems," Memorandum No. UCB/ERL M82/85, College of Engineering, University of California, Berkeley, 1982.

[STON 83] Stonebraker, M. et al. "Performance Enhancements to a Relational Database System," ACM Transactions on Database Systems, Vol. 8, No. 2, June 1983.

[SU 81a] Su, S. et al. "A DMS Cost/Benefit Decision Model: Cost and Preference Parameters," Report NBS-GCR-82-373, National Bureau of Standards, January 1981.

[SU 81b] Su, S. et al. "A DMS Cost/Benefit Decision Model: Analysis, Comparison, and Selection of DBMS's," Report NBS-GCR-82-375, National Bureau of Standards, July 1981.

[TEMP 82] Templeton, M., Kameny, I., Kogan, D., Lund, E., Brill, D., "Evaluation of Ten Data Management Systems", SDC document TM-7817/000/00.

[TEOR 76] Teorey, T. and Das, K. "Application of an Analytical Model to Evaluate Storage Structures," Proceedings of ACM SIGMOD Conference, 1976.

[TUEL 75] Tuel, W. and Rodriguez-Rosell, J. "A Methodology for Evaluation of Data Base Systems," IBM Research

Report RJ 1668, 1975.

[ULLM 82] Ullman, J. Principles of Database Systems, Second Edition, Computer Science Press, 1982.

[WALT 76] Walters, R. "Benchmark Techniques: A Constructive Approach," The Computer Journal, Vol. 19, No. 1, 1976.

[WEIS 81a] Weiss, H. "Down-scaling DBMS to the Microworld," Mini-Micro Systems, April 1981.

[WEIS 81b] Weiss, H. "Which DBMS is Right for You?" Mini-Micro Systems, October 1981.

[YAO 74] Yao, S. "Evaluation and Optimization of File Organizations through Analytic Modeling," PhD Thesis, University of Michigan, 1974.

[YAO 75] Yao, S. and Merten, A. "Selection of File Organizations through Analytic Modeling," Proceedings First VLDB Conference, Framingham, 1975.

[YAO 77a] Yao, S. "An Attribute Based Model for Database Access Cost Analysis," ACM Transactions on Database Systems, Vol. 2, No. 1, 1977.

[YAO 77b] Yao, S. "Approximating Block Accesses in Database Organizations," Communications of the ACM, Vol. 20, No. 4, 1977.

[YAO 78] Yao, S. and DeJong, D. "Evaluation of Database Access Paths," Proceedings of ACM SIGMOD Conference, 1978.

[YAO 79] Yao, S. "Optimization of Query Evaluation Algorithms," ACM Transactions on Database Systems, Vol. 4, No. 2, 1979.

[YAO 84] Yao, S.B., Hevner, A., and Yu, S.T. "Architectural Comparisons of Three Database Systems," Report submitted to the National Bureau of Standards, April 1984.

[YOUS 79] Youseffi, K. and Wong, E. "Query Processing in a Relational Database System," Proceedings Fifth VLDB, 1979.

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. NBS/SP-500/118	2. Performing Organ. Report No.	3. Publication Date December 1984

4. TITLE AND SUBTITLE

Computer Science and Technology:
A Guide to Performance Evaluation of Database Systems

5. AUTHOR(S) Daniel R. Benigni, Editor
Prepared by: S. Bing Yao and Alan R. Hevner

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

Software Systems Technology, Inc.
7100 Baltimore Avenue, Suite 206
College Park, MD 20740

7. Contract/Grant No.

8. Type of Report & Period Covered
Final

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

National Bureau of Standards
Department of Commerce
Gaithersburg, MD 20899

10. SUPPLEMENTARY NOTES Library of Congress Catalog Card Number: 84-601144
Related Documents: NBS-GCR-84-467, Performance Evaluation of Database Systems - A Benchmark Methodology by S. Bing Yao and Alan R. Hevner; NBS-GCR-84-468, An Analysis of Three Database System Architectures Using Benchmarks by S. Bing Yao and Alan R. Hevner

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

This guide presents a generalized performance analysis methodology for the benchmarking of database systems. The methodology identifies criteria to be utilized in the design, execution, and analysis of a database system benchmark. This generalized methodology can apply to most database system designs. In addition, presenting a wide variety of possible considerations in the design and implementation of the benchmark, this methodology can be applied to the evaluation of either a single system with several configurations, or to the comparison of several systems.

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

benchmark execution; benchmark methodology; benchmark workload; database systems; DBMS; indexing; performance evaluation; query complexity; response time.

13. AVAILABILITY

☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☒ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

☐ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES
54

15. Price

USCOMM-DC 6043-P80

ANNOUNCEMENT OF NEW PUBLICATIONS ON COMPUTER SCIENCE & TECHNOLOGY

Superintendent of Documents,
Government Printing Office,
Washington, DC 20402

Dear Sir:

Please add my name to the announcement list of new publications to be issued in the series: National Bureau of Standards Special Publication 500-.

Name _____

Company _____

Address _____

City _____ State _____ Zip Code _____

(Notification key N-503)

*U.S. GOVERNMENT PRINTING OFFICE : 1984 0-461-105/10155

NBS Technical Publications

Periodical

Journal of Research—The Journal of Research of the National Bureau of Standards reports NBS research and development in those disciplines of the physical and engineering sciences in which the Bureau is active. These include physics, chemistry, engineering, mathematics, and computer sciences. Papers cover a broad range of subjects, with major emphasis on measurement methodology and the basic technology underlying standardization. Also included from time to time are survey articles on topics closely related to the Bureau's technical and scientific programs. As a special service to subscribers each issue contains complete citations to all recent Bureau publications in both NBS and non-NBS media. Issued six times a year.

Nonperiodicals

Monographs—Major contributions to the technical literature on various subjects related to the Bureau's scientific and technical activities.

Handbooks—Recommended codes of engineering and industrial practice (including safety codes) developed in cooperation with interested industries, professional organizations, and regulatory bodies.

Special Publications—Include proceedings of conferences sponsored by NBS, NBS annual reports, and other special publications appropriate to this grouping such as wall charts, pocket cards, and bibliographies.

Applied Mathematics Series—Mathematical tables, manuals, and studies of special interest to physicists, engineers, chemists, biologists, mathematicians, computer programmers, and others engaged in scientific and technical work.

National Standard Reference Data Series—Provides quantitative data on the physical and chemical properties of materials, compiled from the world's literature and critically evaluated. Developed under a worldwide program coordinated by NBS under the authority of the National Standard Data Act (Public Law 90-396). NOTE: The Journal of Physical and Chemical Reference Data (JPCRD) is published quarterly for NBS by the American Chemical Society (ACS) and the American Institute of Physics (AIP). Subscriptions, reprints, and supplements are available from ACS, 1155 Sixteenth St., NW, Washington, DC 20056.

Building Science Series—Disseminates technical information developed at the Bureau on building materials, components, systems, and whole structures. The series presents research results, test methods, and performance criteria related to the structural and environmental functions and the durability and safety characteristics of building elements and systems.

Technical Notes—Studies or reports which are complete in themselves but restrictive in their treatment of a subject. Analogous to monographs but not so comprehensive in scope or definitive in treatment of the subject area. Often serve as a vehicle for final reports of work performed at NBS under the sponsorship of other government agencies.

Voluntary Product Standards—Developed under procedures published by the Department of Commerce in Part 10, Title 15, of the Code of Federal Regulations. The standards establish nationally recognized requirements for products, and provide all concerned interests with a basis for common understanding of the characteristics of the products. NBS administers this program as a supplement to the activities of the private sector standardizing organizations.

Consumer Information Series—Practical information, based on NBS research and experience, covering areas of interest to the consumer. Easily understandable language and illustrations provide useful background knowledge for shopping in today's technological marketplace.

Order the **above** *NBS publications from: Superintendent of Documents, Government Printing Office, Washington, DC 20402.*

Order the **following** *NBS publications—FIPS and NBSIR's—from the National Technical Information Service, Springfield, VA 22161.*

Federal Information Processing Standards Publications (FIPS PUB)—Publications in this series collectively constitute the Federal Information Processing Standards Register. The Register serves as the official source of information in the Federal Government regarding standards issued by NBS pursuant to the Federal Property and Administrative Services Act of 1949 as amended, Public Law 89-306 (79 Stat. 1127), and as implemented by Executive Order 11717 (38 FR 12315, dated May 11, 1973) and Part 6 of Title 15 CFR (Code of Federal Regulations).

NBS Interagency Reports (NBSIR)—A special series of interim or final reports on work performed by NBS for outside sponsors (both government and non-government). In general, initial distribution is handled by the sponsor; public distribution is by the National Technical Information Service, Springfield, VA 22161, in paper copy or microfiche form.